

Master's Thesis

Statistical Analysis of Chao's Immune System Simulation

Eric Winnington
eric.winnington@epfl.ch

Advisors:
Prof. Jean-Yves Le Boudec
Slavisa Sarafijanovic

2005

LCA - I&C

Contents

1	Introduction	4
2	State of the Art	5
3	Markov Model extraction	5
3.1	Markov Chain	5
3.2	Markov Process	9
4	Statistical validation of the simulation	10
4.1	Time step verification	11
4.2	Validating the Random number generator	13
4.3	Confidence intervals	14
4.4	Testing simulator output: Secondary Reaction Occurs	14
4.5	Impact of Match rule on secondary response	16
5	Conclusion	16
A	Markov Model Equations	19
A.1	Constants & Laws	20
A.2	Without Exhaustion	21
A.3	With Exhaustion	22
B	Markov Process	24
B.1	Introduction	24
B.2	Parameters	24
B.3	Parallel equations	25
C	NI simulator without T-cell interaction	26
D	Simulator initialization sequence	28

Abstract

Modeling the immune system (IS) means putting together a set of assumptions about its components (cells and organs) and their interactions. Simulations of a model show joint behavior of the components, which for complex realistic models is often impossible to find analytically. Simulations allow us to experiment on how initial concentrations and properties of the immune cells and viruses impact the IS behavior, and gain better quantitative and qualitative insight into how the IS works and why different behavior patterns occur.

A simulation, once it has been created, must be reviewed both statistically and analytically as well as validated from the biological point of view.

We analyzed Chao's immune system simulation [1][2] from a statistical and analytical point. We explicited both the Markov chain which was simulated and the underlying process on which Chao's stage-structured approach was built. Furthermore, we established a test protocol for timestep validation which Chao's simulator passed. We evaluated Chao's simulator's dependence on the random number generator, which was shown to be negligible.

Finally, we evaluated the simulator output and our major result is the discovery of a secondary response to a primary infection, an occurrence is not shown in Chao's dissertation. A tertiary response to the infection is never possible due to the size of the secondary response caused by memory cells.

1 Introduction

The Immune System (IS) is a highly complex and immensely adaptable mechanism which appears in all life forms from bacteria upwards. The IS comprises two parts: the innate IS which protects the body from a large amount of pathogens using defenses that are quickly mobilized and triggered by receptors that recognize a broad spectrum of pathogens; and the adaptive IS which develops specific cells to combat infections by viral agents. Our understanding of the mechanism of the IS is primordial for progress in medical science.

Cytotoxic T lymphocytes (CTL), also known as killer T-cells, are potent weapons of the adaptive IS for combating viral infections. They destroy infected cells by forcing them to commit suicide (apoptosis). To recognize infected cells, they scan the MHC molecules on the surface of the cell and detect non-self peptides. Once a molecule has been recognized by the T-cell Receptor Chains that are on the surface of the CTL, a vesicle containing granzyme and perforin is inserted into the cell, forcing apoptosis. A CTL can also kill a cell by using Fas Ligand to bind the Fas protein on the surface of the target cell, again causing apoptosis.

Although in vivo experiments are primordial for the understanding of the IS, they are usually very costly and can take months for an experienced team to perform. They are also very complex to follow since it is difficult to observe them without disrupting the ongoing process. So, in vivo experiments can usefully be complemented by a mathematical and statistical approach. Computer Science can help and enhance our understanding of the IS by allowing us to simulate complex systems and behaviors that cannot be observed directly. Computer models can simulate experiments lasting years in a far shorter time and give access to information which is normally hidden during in vivo studies. The precision of these simulations depends on the model used to represent the IS.

Once a model has been established, it needs to be validated conceptually and numerically. This means that each part of the simulator must be evaluated and appreciated against current knowledge. This enables us to refine and improve the simulation through constructive criticism.

Dennis Lai Chao proposed an IS simulator in [1][2] which describes the CTL response to a viral infection.

We analyzed Chao's model and formulated the Markov chain which it simulated but did not explicitly define in [2]. We extracted the underlying Markov process which is approximated by the Markov chain. We established a timestep validation protocol which Chao's simulator passed. We replaced the Random Number Generator (RNG) with the higher performing Mersenne Twister [7] RNG in order to investigate how RNG dependent the simulation is. Finally, we evaluated the Chao's simulator output and discovered that a primary infection could cause a secondary T-cell response. This secondary T-cell response is consistent with what would happen in case of reinfection by the same virus. This was not

mentioned in Chao’s work but anyone using his simulator should be aware of this response.

2 State of the Art

Research into IS simulations is currently ongoing using several approaches. Chao’s [1][2] approach, which we analyzed, is to use a stage-structured model with a minimal number of modeled populations to enable rapid simulations and a concise overview of the immune response.

ParImm [3] is an IS simulator which uses a cellular automata based model to simulate small scale systems to generate a quantitative picture of the IS. This approach is demanding both in terms of memory and processing power.

Mathematical models have been developed to explain and predict part of the IS behavior as in [4] which deals with the size of the memory T-cell repertoire by modeling it as several interacting compartments or in [5] which proposes a model for T-cell proliferation in the presence of antigen but does not deal with antigen recognition by T-cells. This recognition has been modeled in [6] which focuses on the specific problem of TCRs and their binding, and resolves the problem by using activation curves to model the activation properties of a T-cell repertoire.

3 Markov Model extraction

In Chao’s papers, the simulated IS model is explained but never rigorously defined. This is a problem when it comes to reviewing the model and validating it. Chao’s stage-structured approach is an approximation of an underlying Markov process. Our first step was to extract the discrete time Markov model, the Markov chain, which Chao simulates as his stage-structured model.

A Markov chain is a discrete stochastic process with the Markov property. It is a sequence of X_1, X_2, X_3, \dots of random variables. The range of these variables is called the state space S , the value of X_n being the state of the process at time n . If the conditional probability distribution of X_{n+1} on past states is a function of X_n alone, then:

$$P(X_{n+1} = x | X_0, X_1, X_2, \dots, X_n) = P(X_{n+1} = x | X_n)$$

where x is some state of the process. The identity above identifies the Markov property. Since Chao’s simulation depends only on its current state to calculate the future state, we can express it as a Markov chain.

3.1 Markov Chain

The first task is to define the state space S of the Markov chain. S is composed of all the possible populations to which the simulation components can belong in

the course of the simulation. In Chao's case, we define the following populations: T target cells, I infected cells, V virus particles, N_T naive T-cells, M_T memory T-cells, A_T^j activated T-cells, A_{MT}^k activated memory T-cells, E_{TA}^i effector T-cells waiting to divide, E_{TB}^{il} effector T-cells dividing, E_{MTA}^i memory effector T-cells waiting to divide, E_{MTB}^{il} memory effector T-cells dividing and W_{MT}^m T-cells waiting to become memory cells. In addition, it is useful to define a last population that is not part of the state but serves to simplify formulas: E_* the sum of all effector T-cells. To accurately model Chao's simulation, we introduced the idea of time promotion into the Markov chain. See Fig. 1 for an illustration of a time promoted Markov state. Several times during the simulated lifetime of a t-cell there are steps where it has to wait in a certain state for a number of timesteps. To model this, these states are superscripted with a index which defines the point in the process the T-cell is at. In these indexes: i represents the generation of the T-cell, j represents the activation delay for a naive T-cell, k represents the activation delay for memory T-cells, l represents the division delay for effector T-cells and m represents the delay for a T-cell to convert to memory. These states are summarized in Table 1. Finally, we can represent

$$S = (T, I, V, N_T, M_T, A_T^j, A_{MT}^k, E_{TA}^i, E_{MTA}^i, E_{TB}^{il}, E_{MTB}^{il}, W_{MT}^m) \quad (1)$$

See fig. 2 for a representation of the states and their interactions and see fig. 3 for a close-up of the T-cell cycle.

Population	Description	Index Value
T	Target cells	
I	Infected cells	
V	Virus Particles	
N_T	Naive T-cells	
M_T	Memory T-cells	
A_T^j	Activated T-cells	19 hours
A_{MT}^k	Activated Memory T-cells	1 hour
E_{TA}^i	Effector T-cells waiting to divide	18 generations
E_{TB}^{il}	Effector T-cells dividing	18 gen, 5 hours
E_{MTA}^i	Effector Memory T-cells waiting to divide	18 generations
E_{MTB}^{il}	Effector Memory T-cells dividing	18 gen, 5 hours
W_{MT}^m	T-cells converting to memory	14 days
E_*	Sum of all Effector Cell populations	

Table 1: Populations composing the state space S

The Markov chain transition formulas and a model recapitulation can be found in App. A. A Markov chain that includes Exhaustion as defined by Chao

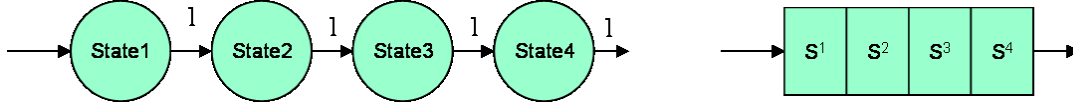


Figure 1: A 4 state time promoted Markov chain in which an item goes through the states advancing at each timestep (left) will be represented as shown on the right.

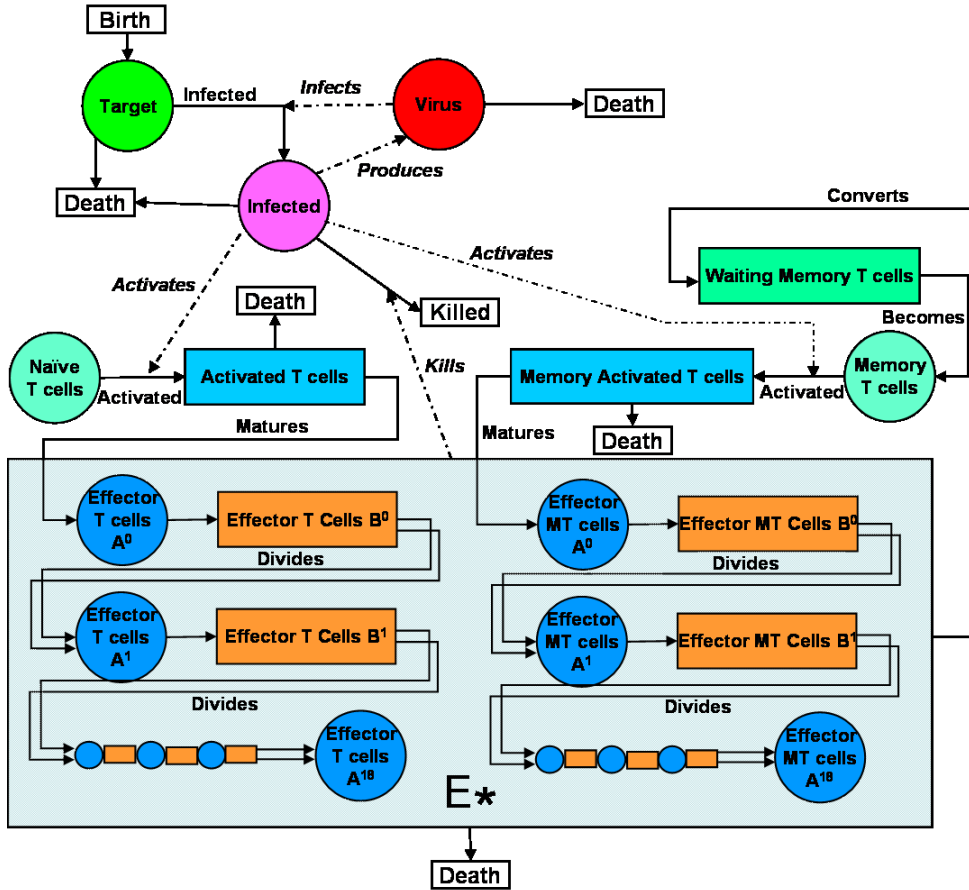


Figure 2: These are the relations between the different populations of the IS simulator. Transitions between populations are marked by a full line whereas effects are denoted by a dotted line. Circles are representations of populations and boxes are time promoted populations. Here we see how a Target cell population is infected by viruses to become infected cells which produce virus particles. The only impact of the simulated T-cells is to reduce the number of infected cells by killing them. All the other effects of the immune system are subsumed in two effects: the high death rate of the virus particles and the high death rate of infected cells. The E_* population is shown as a shaded rectangle.

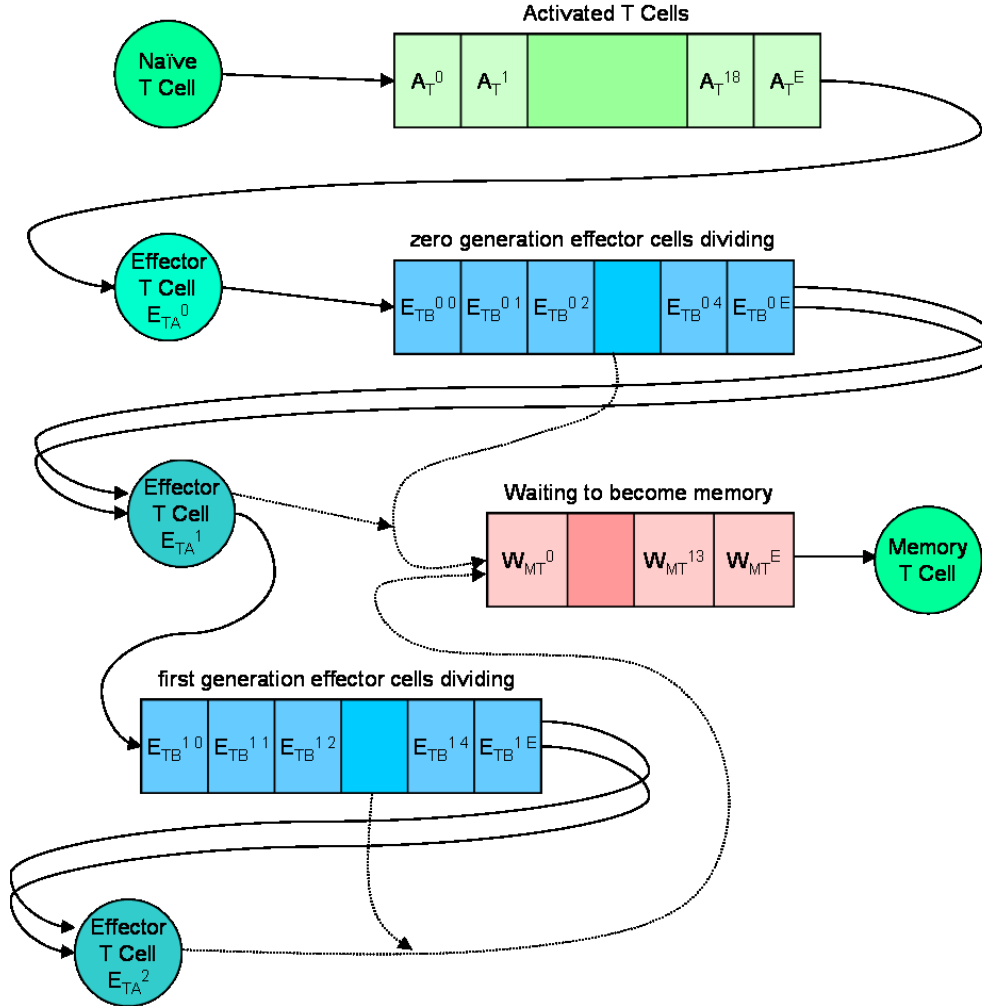


Figure 3: Graphical representation of the naive T-cell cycle in the simulation. The naive T-cell is first activated and progresses through activation to become an effector. An effector cell passes from state A (waiting to divide) to state B (dividing) before returning two copies of itself to state A. This cycle continues for 18 generations. At each step, an effector T-cell can be recruited and start to convert to a memory cell. Not shown in this flowchart is the fact that once the cell has been activated, it is possible for it to die at any timestep.

was also derived; the modifications to be applied to the first model can be found in App. A.3.

Once this model was established we were able to analyze it and extract the underlying continuous time process of which the Markov chain was an approximation.

3.2 Markov Process

A Markov process is a continuous time stochastic process which takes into account not only the change of state but also the actual time of the transitions. In our Markov chain model we approximated time delays with a succession of states; in our Markov process we can directly use the time delays since we are working on continuous time.

Chao’s stage-structured approach is an approximation of an underlying Markov process. His assumptions are that during a timestep the populations stay constant and that the transition probabilities are independent or that these factors have so small an impact as to be negligible.

Origin	Parameter	Effect	Transition Rate	
	$\xrightarrow{\lambda}$	$T + 1, I, V$	λ	$T_0 = 10^6$
T	$\xrightarrow{\delta_T}$	$T - 1, I, V$	$\delta_T \cdot T$	$I_0 = 0$
TV	$\xrightarrow{\beta}$	$T - 1, I + 1, V$	$\beta \cdot V \cdot T$	$V_0 = 50$
I	$\xrightarrow{\delta_I}$	$T, I - 1, V$	$\delta_I \cdot I$	$\lambda = 5 \cdot 10^4$
I	$\xrightarrow{\pi}$	$T, I, V + 1$	$\pi \cdot I$	$\delta_T = 0.01$
V	\xrightarrow{c}	$T, I, V - 1$	$c \cdot V$	$\beta = 2 \cdot 10^{-7}$
				$\delta_i = 0.7$
				$\pi = 100$
				$c = 2.3$

Table 2: Chao’s partial continuous model expressed as a set of parallel equations from [2][S3.1 p26] which he uses to validate his choice of timestep.

In Chao’s thesis [2][S3.1], a simplified continuous time model without T-cell interaction is rigorously defined as the basis of his modelization. He uses it to validate his choice of timestep. We built on this base model (repeated as Table 2 for reference) and on our Markov Chain (Section 3.1) to create a continuous time Markov Process which is the real representation of what Chao simulates without any sampling bias from the timestep.

In our Markov Process, we make use of a scheduler to enable exact time delays. A cell entering at time t a time promotion state of delay d is entered into the scheduler with an exit time of $t + d$. During this time, the cell can still die or be recruited to convert to memory (if it is an effector cell). At the specified time $t + d$, the cell will exit the time promotion state and continue to the next state. A graphical representation of our Markov process can be found in fig. 4.

Our full model can be found in App. B. This model is the underlying Markov process that should be verified by immunologists to discover if any inappropriate transitions occur.

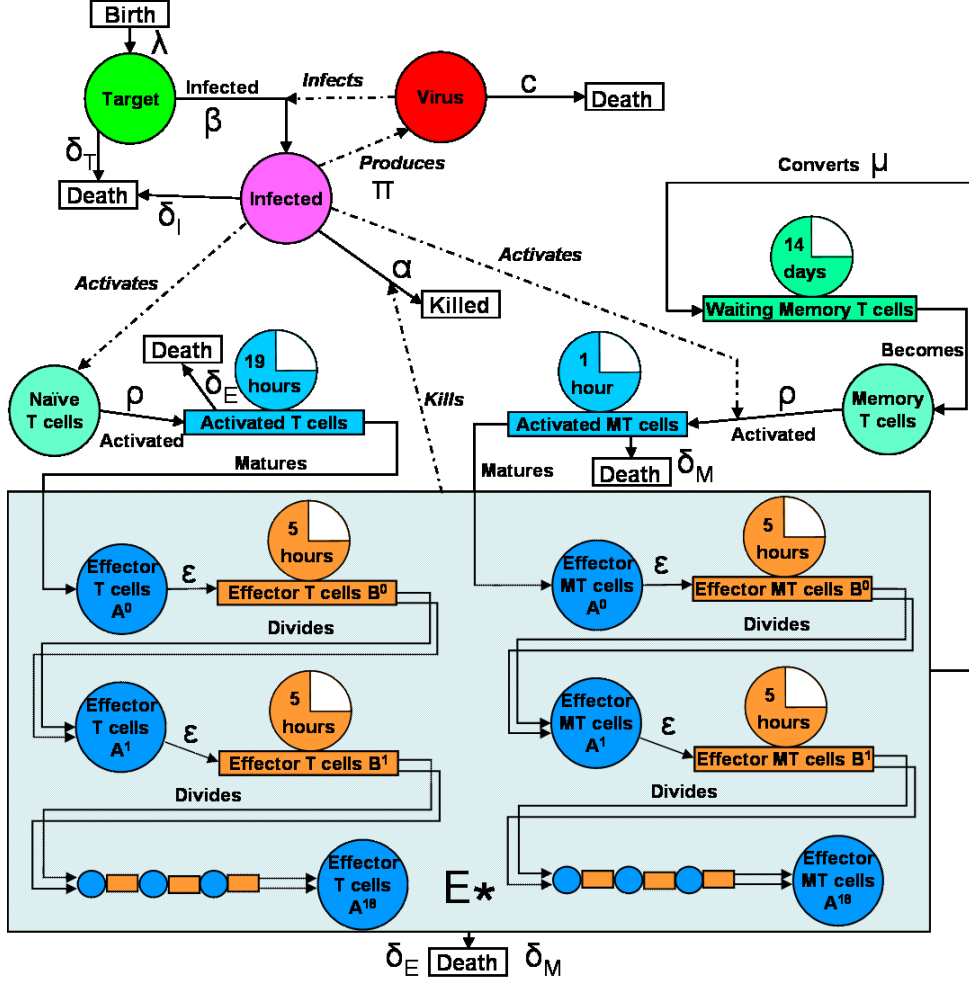


Figure 4: Illustration of the underlying Markov Process with transition rates. The scheduler delay is represented with a clock above the state, inside of which we see the duration of this delay. The E_* population is shown as a shaded box.

4 Statistical validation of the simulation

Once the models have been extracted from the simulation, we can begin to analyze the simulator itself. A simulation should be independent from the timescale at which it is run, or at least the results should only be more precise as the timesteps diminish towards zero. It is expected that with larger timesteps results

can diverge due to approximation errors. Our first test will then be to verify the behavior of the simulator on a large range of timesteps. The Random Number Generator (RNG) contributes to the simulation’s quality; Sub-random bits will skew results and can cause false output. We will test several RNGs and plot simulation results to compare them. The simulator’s output will be processed and shown to discover any rare events or inappropriate behavior. We have used three simulators for these tests: Chao’s original code (CO) without any modification, Chao’s code modified (CM) to include the Mersenne Twister RNG [7] and the CERN’s COLT library [9] and a new implementation (NI) of the model without T-cell interaction. The NI simulator can be found in Appendix C.

4.1 Time step verification

Our first analysis was to verify the scaling of all time-dependant equations in the simulator, calculating the expected value and the value used by the simulator. We did not detect any error in the scaling of the equations. The scaled formulas take the following form

$$1.0 - e^{(-x/T)} \quad \text{with } x \text{ parameter and } T \text{ constant timesteps in one day} \quad (2)$$

To validate the 10 minute timesteps which Chao used in his simulation, we ran 100 simulations without T-cell interaction each on a twelve value range of timesteps (60 minutes down to 1 minute) for each simulator (CO, CM and NI). We also ran the exact stochastic simulations provided by Chao in the form of the Gillespie Direct Method and the Gillespie First Reaction simulation method. This enables us to compare continuous processes like the Gillespie simulations to discrete chain simulations.

Our results indicate that the output is independent of the timestep chosen, which is what is mathematically expected. Shown in figure 5 are the results for the Gillespie Direct Method, the Gillespie First Reaction, CO at 10 and 3 minute timesteps and CM at 10 and 3 minute timesteps. All three simulators perform identically and their output matches the Gillespie simulations, with a better matching as the timesteps get shorter, but with a longer run time. Our range of timestep values was 60, 30, 20, 15, 12, 10, 6, 5, 4, 3, 2 and 1 minute.

We tested the whole range of timestep values and conclude that the 10 minute timestep chosen by Chao provides the best tradeoff point between result precision and execution time. For larger timescales, the estimations cause a slight deviation from expected values, whereas for smaller timescales the gain in precision is not worth the extended run time. To conclude, we validate the 10 minute timestep that Chao used in his simulator.

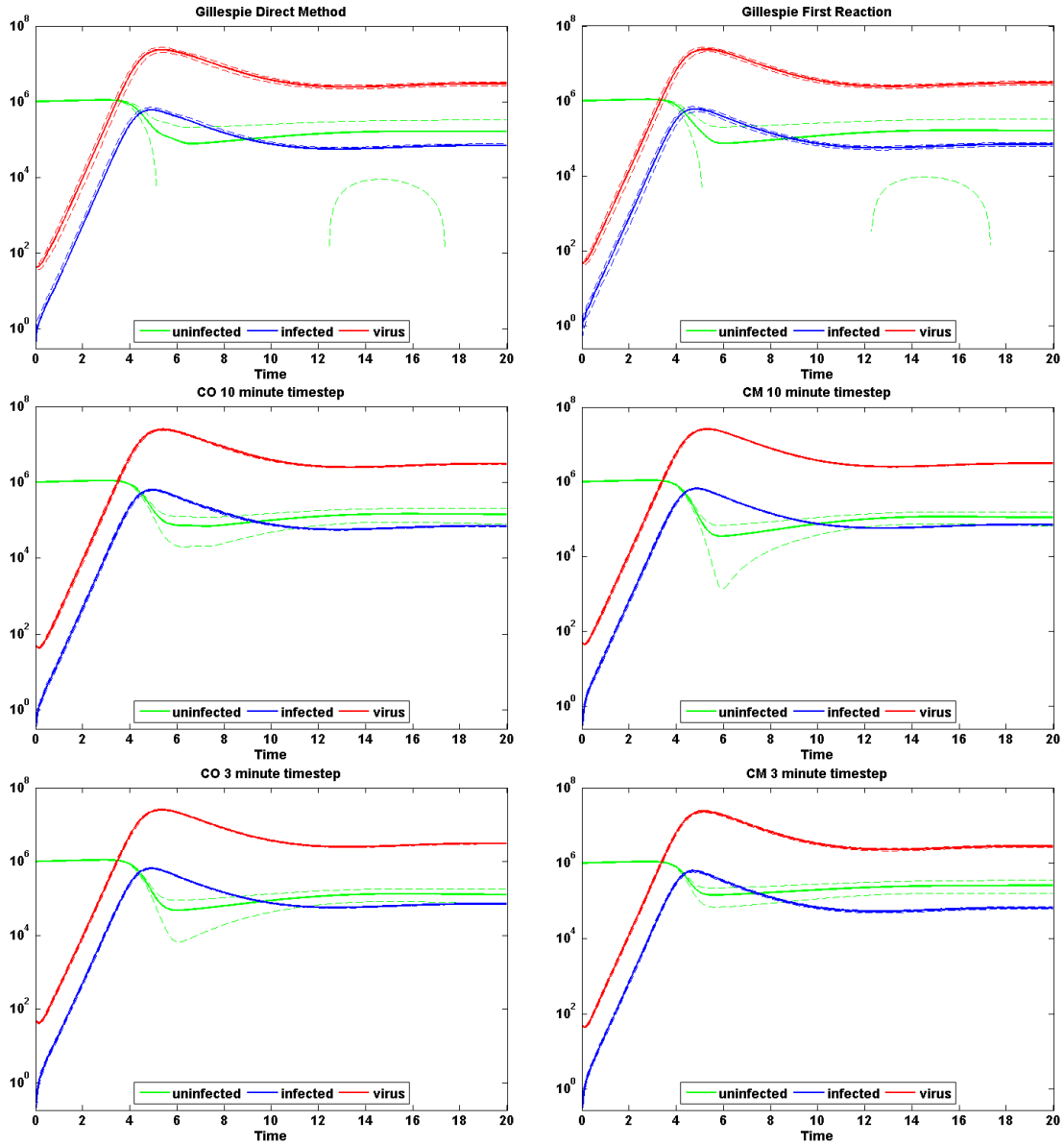


Figure 5: Dynamics of model without T-cell interaction. X-axis represents time in days and Y-axis represents population size. All graphs are plotted with a 5% confidence interval. Represented in green is the target cell population, in blue the infected cell population and in red the viral population. We can see how all the simulations comply with the Gillespie results confirming that the simulation is independent of the timestep.

4.2 Validating the Random number generator

In simulations, the quality of the random number generator is primordial. A bad random number generator will produce sub-random bits which skew the simulation. The Generator needs to have a long period to ensure that a cycle of random numbers never occurs. It is also necessary for the random numbers to be appropriately distributed in the entire space.

Chao uses an adapted version of ran3 from Numerical Recipes in C, an algorithm based on Knuth’s subtractive method. Ran3 is reported [8] to have a period of $2^{55}-1$ and is known to have significant correlations on the bit level. We used his implementation in CO.

For comparison, the Mersenne Twister [7] was used in CM. Its period is known to be $2^{19937}-1$, and it has an assured 623-dimensional equidistribution property. Because of its efficiency the Mersenne Twister is utilized in many packages including the CERN’s COLT [9] distribution. We also added the TT800 random number generator which is the ancestor of the Mersenne Twister and the two 16-bit multiply with carry generator which is one of the simplest generators to acceptably pass the DIEHARD test.

We used the Marsaglia DIEHARD [10] battery of tests to determine the randomness of the output from these two generators. Each of the Diehard tests produced one or more p-values. We categorized the p-values as good, suspect, or rejected, as in [8]. We classified a p-value as rejected if $p > 0.998$. We classified a p-value as suspect if $0.95 \leq p \leq 0.998$. We classified all other p-values as good. We assigned point values to the two Random Number Generators: two points for every reject classification, one point for every suspect classification, and no points for every good classification. Then, we summed these points to produce a final Diehard score for each. Low Diehard scores indicate good quality, whereas high Diehard scores indicate poor quality.

A visual comparison of the results can be seen in figure 6. We recommend that the Mersenne Twister be used due to its very large proven period; it will not cause cycling trouble in large simulations. Changing the random number generator showed that the simulation was not RNG dependant. Refer to figure 5 for a comparison of CO and CM results.

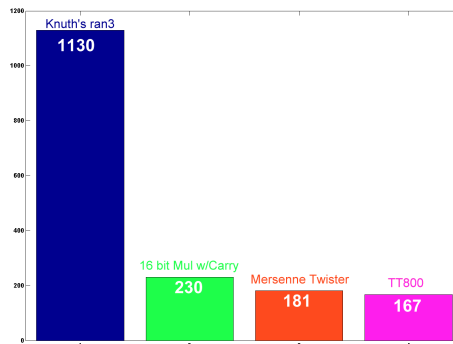


Figure 6: Total scores of the 4 different generators for 10 DIEHARD runs. A high value indicates sub-random bits whereas a low value indicates that the bits are randomly distributed.

4.3 Confidence intervals

None of the results Chao gives from his simulator include confidence intervals (CI). CIs are calculated from a sufficiently large number of runs which we assume are independent and normally distributed. For a 5% confidence interval we calculate the 0.975 quantile of the Student distribution with $N-1$ degrees of freedom. This confidence interval is distributed around the mean value of the runs. We included confidence intervals on all our measurements of the simulation and they validate that our outputs are consistently within acceptable values.

4.4 Testing simulator output: Secondary Reaction Occurs

Having analyzed the simulator, we started to reproduce graphs from Chao's papers. This is to verify that what is reported in the papers is a representation of what can be found by executing the IS simulator. The run parameters we used were 1000 and 500 initial virus particles using the hamming match rule and no subsequent injections of virus. The simulations were also run with and without the T-cell exhaustion model defined in his thesis[2]. While we were analyzing the results we discovered events that were not reported in Chao's papers. These occur with a non negligible frequency and show a resurgence of the virus after the body fails to remove it due to the primary T-cell response. When T-cell exhaustion is added to the model, the non-clearance rate after a primary reaction soars and in some cases the virus is not cleared in 50 full simulation days. Both CO and CM simulators expressed this behavior, leading us to conclude that this was not an artifact of the RNG or the Poisson and Binomial evaluators.

The results for the 1000 initial virus concentration, using a Hamming match rule with no exhaustion, as show in Fig. 7, demonstrate this lack of clearance after the primary reaction. The virus then is able to subsist at low levels for several days (between days 14 and 17) until the effector cells are too low to kill the infected cells faster than the virus can infect them. At that point, the virus can again multiply unchecked until the memory cells created in the primary reaction become themselves effectors and eradicate the virus with a secondary effector T-cell response. At the same time, the high affinity naive cells which were present to respond to the first infection have been depleted; consequently they are not available to help combat the resurgence of the virus.

Runs made with no conversion of effector T-cells to memory confirm that if there is a resurgence of the virus, it is necessary to have memory cells to initiate a secondary reaction. No memory cells and depleted naive cells spells death for the organism.

What is not clear is whether these secondary reactions are normal IS behavior in which a virus subsists at a low level in the organism before returning, or whether these results are an artifact introduced by inappropriate modeling.

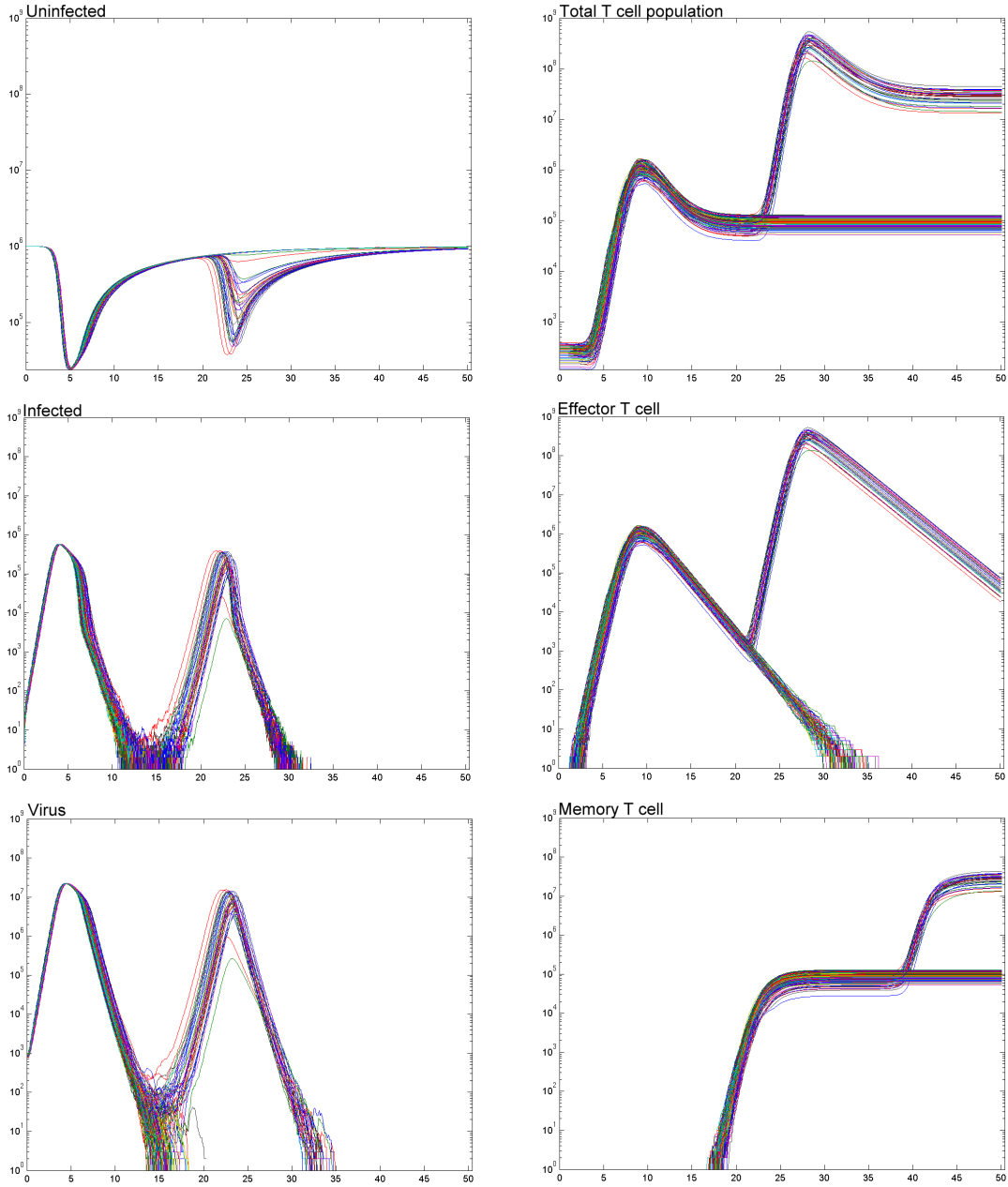


Figure 7: Results for the 1000 virus, Hamming match rule without exhaustion runs. All the runs are plotted for ease of comparison. The x axis is the time expressed in days. The results show the failure to clear the infection with the primary T-cell reaction, forcing a second reaction which is supported by memory cells created in the primary reaction. In almost a quarter of the runs (22.5%), this secondary reaction is necessary to remove all the virus from the organism.

4.5 Impact of Match rule on secondary response

To determine the impact of the match rule on the secondary response probability, we ran simulations with all three of the match rules that Chao defined in his research, namely Xor, Manhattan and Hamming. The virus loads were 1000 and 500 as before and we tested both with and without exhaustion.

Primary response clearance rates			
Matching Rule	Exhaustion	500 Virus	1000 Virus
Hamming	Without	15%	22.5%
	With	98%	96.5%
Manhattan	Without	31.7%	31.5%
	With	100%	99%
Xor	Without	31%	37%
	With	100%	100%

As Chao explains in his thesis, the Manhattan and Xor rules produce results consistent with each other, while the Hamming distance yields different results. But all three match rules continue to produce a secondary reaction, although for Hamming the secondary reaction probability is much lower.

5 Conclusion

In [1] and [2], Chao did not explicitly define the Markov chain which he was simulating. We expressed it and the underlying Markov process which can now be reviewed by immunologists to improve and expand its scope.

His simulator passed our validation tests on timescaling, showing that the simulation results are consistent across timesteps. We also validated Chao's choice of a 10 minute timestep as a good tradeoff between precision and simulation speed. We also showed that the simulation was Random Number Generator independent and that changing it had little influence in the results, although ran3 performed worse than the Mersenne Twister in the Marsaglia's DIEHARD battery of tests.

The secondary T-cell reaction seen in the simulator's output was described and there is a need to review the biological significance of this response caused by the non-clearance of viral particles at the end of the primary reaction. This secondary reaction is caused by effector T-cells which only exist within a short timeframe (around 30 days) and are only in sufficient concentration to effectively combat the virus for about 15 days. During this time, they need to remove all infected cells from the organism and last long enough to outlive the viral particles still remaining. Secondary reactions occur when the virus survives longer than this timeframe and is then left to reinfect the organism once the effectors have disappeared. This causes a resurgence of the virus and activates memory cells created during the primary reaction. The memory effector T-cell response is

much larger than the naive response due to the number of clones that compose it and to its smaller death rate. This is sufficient to finally remove all virus from the organism.

The secondary reaction that occurs in Chao's simulation should have been reported in his thesis. All three different match rules showed this secondary T-cell reaction, but it is not mentioned or analyzed by Chao. This brings us to the question: Is this reaction normal IS behavior or is it an artifact of the model? This question will have to be answered before this simulator can be used for studying IS responses.

To conclude, mathematical modeling of the IS is useful because by defining the model we understand the IS interactions and express them removing what is irrelevant and redundant. These models allow us to simulate test scenarios which are difficult to implement in in-vivo research. The results of the simulations give us new ideas about virus dynamics and can give insight into new therapeutical solutions.

References

- [1] D. L. Chao: *A stochastic model of cytotoxic T-cell responses*, J. theor. Biol. 228 (2004), 227-240.
- [2] D. L. Chao: *Modeling the cytotoxic T-cell response*, PhD diss. University of New Mexico, 2004.
- [3] M. Bernaschi and F. Castiglione: *Design and Implementation of an Immune System Simulator*, Computers in Biology Medicine Vol. 31 (2001) 303.
- [4] R. E. Callard: *Fratricide: a mechanism for T memory-cell homeostasis*, TRENDS in Immunology Vol.24 No.7 July 2003.
- [5] R. J. De Boer, A. S. Perelson: *Towards a General Function Describing T-cell Proliferation*, J. theor. Biol. 175 (1995), 567-576.
- [6] H. A. Van Den Berg: *A reliable and Safe T-cell Repertoire based on Low-affinity T-cell Receptors*, J. theor. Biol. 209 (2001), 465-486.
- [7] M. Matsumoto, T. Nishimura, *Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator*, ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, January pp.3-30 (1998).
- [8] B. M. Gammel: *Hurst's rescaled range statistical analysis for pseudorandom number generators used in physical simulations*, Phys.Rev.E58 2586-2597 (1998).

- [9] The Colt Distribution, Open Source Libraries for High Performance Scientific and Technical Computing in Java available at <http://hoschek.home.cern.ch/hoschek/colt/>
- [10] The Marsaglia Diehard Battery of tests of randomness available at <http://stat.fsu.edu/pub/diehard/>

A Markov Model Equations

$$S = (T, I, V, N_T, M_T, A_T^j, A_{MT}^k, E_{TA}^i, E_{MTA}^i, E_{TB}^{il}, E_{MTB}^{il}, W_{MT}^m)$$

$$E_* = \sum E_{TA}^i + \sum E_{MTA}^i + \sum E_{TB}^{il} + \sum E_{MTB}^{il}$$

with $i = 0 \dots 18$ number of times a T-Cell can divide

$j = 0 \dots (19 \cdot tsph) - 1$ delay from the activation of a T-cell to effectiveness

$k = 0 \dots tsph - 1$ delay from activation of a memory T-cell to effectiveness

$l = 0 \dots (5 \cdot tsph) - 1$ delay before cell division finishes

$m = 0 \dots (14 \cdot tspd) - 1$ delay before cells converting to memory become inactivated memory T-cells

E end of array

$tsph$ and $tspd$ refer respectively to timesteps per hour and timesteps per day. They are for the standard time scale of 10 minutes, 6 and 144 respectively.

T Target T-cell population. Has a constant growth rate of $5 \cdot 10^4$ and a death rate of 0.01 when no T-cell is simulated and a 10^5 growth and 0.1 death when T-cells are simulated, is also reduced when cells are infected by virus.

I Infected cell population. Growth is determined by the number of target cells infected, death rate is 0.7 or 0.8 (depending on presence of T-cells) and population is diminished by effective T-cells.

V Virus population. Represents free virus particles in the body. Growth is linear to infected cell population and death rate is 2.3.

N_T Naive T-cell population. No growth or death rates. Population diminishes with stimulated recruitment. Chao uses 50 or 10 as initial population depending on whether he simulates one clone or all clones reacting respectively

M_T Inactivated Memory T-cell population. Growth is assured by effective T-cells converting to memory. Contrary to what Chao claims in his thesis, recruitment does not start from the 5th generation of cells, it starts as soon as the cell starts to become effective.

A_T Activated T-cell population. Death rate is 0.6 and recruitment is from naive T-cells. Cells in this state have to wait for 19 hours before becoming effectors.

- A_{MT} Activated Memory T-cell population. Death rate is 0.4 and recruitment is from Memory T-cells. Cells in this state have to wait for 1 hour before becoming memory effectors.
- E_{TA} Phase-A Effective T-cell population. The cells are waiting to start to replicate with a probability of 1 $hour^{-1}$. Death rate is 0.6 and recruitment is from Activated T-cells.
- E_{MTA} Phase-A Effective Memory T-cell population. The cells are waiting to start to replicate with a probability of 1 $hour^{-1}$. Death rate is 0.4 and recruitment is from Activated Memory T-cells.
- E_{TB} Phase-B Effective T-cell population. The cells are in the process of replicating and remain in this state for 5 hours before reverting to Phase A. Death rate is 0.6.
- E_{MTB} Phase-B Effective Memory T-cell population. The cells are in the process of replicating and remain in this state for 5 hours before reverting to Phase A. Death rate is 0.4.
- W_{MT} T-cells converting to memory population. Growth is assured by effective T-cell recruitment 0.02. No death rate. Population can diminish if a chronic simulation is done. Cells take 14 days to pass through this state to become memory T-cells.
- E_* Sum of all effector cells. This is not an actual population; it serves only to simplify the formulae.

A.1 Constants & Laws

From Chao

This groups all the constants used in the IS simulator.

Initial Target cell population 10^6 cells

Target cell birth $5 \cdot 10^4$ per day without T-cell interaction, 10^5 with

Target cell death 0.01 per day without T-cell interaction, 0.1 with

Infectivity $2 \cdot 10^{-7}$ for a fast virus, 10^{-7} for a slow virus

Production Rate 100 for a fast virus, 65 for a slow virus

Infected cell death 0.7 per day without T-cell interaction, 0.8 with

Viral death 2.3 per day
 T-cell effector death 0.6 per day
 Memory T-cell effector death 0.4 per day
 Memory T-cell death 0.0
 Conversion to memory 0.02
 Hours before effective T-cell 19
 Hours before effective Memory T-cell 1
 Times a T-cell can divide 18
 B Phase delay (Bd) 5 hours
 Cycle time (Ct) 6 hours
 Recruitment rate 1 per day
 Time step (TS) 10 minutes
 Time steps per hour (TSPH) $60/TS$
 Time steps per day (TSPD) $24/TSPH$
 Naive to effector (NtE) 1.0
 Peptide Levels (Pl) 1.0
 Affinity (depends on match rule)
 T-cell clearance speed (Tclear) 12
 T-cell naive population 50 for one clone, 10 when all clones are created, $5 \cdot 10^4$ when one big clone is selected

Laws

$P(\lambda)$ Poisson law with parameter λ

$B(n, p)$ Binomial law with parameters n and p , number of tries and probability, respectively

For all the equations, the following assumptions were made: only 1 T-cell clone, 1 virus, 1 epitope for the virus, no viral mutation, the T-cell selection is prior to the model and no exhaustion is implemented.

A.2 Without Exhaustion

$$\begin{aligned}
 T_{t+1} &= T_t + P(10^5/TSPD) - B(T_t, 1.0 - e^{-0.1/TSPD}) - B(T_t, 1.0 - e^{(-2 \cdot 10^{-7}/TSPD) \cdot V_t}) \\
 I_{t+1} &= I_t + B(T_t, 1.0 - e^{(-2 \cdot 10^{-7}/TSPD) \cdot V_t}) - B(I_t, 1.0 - e^{-0.8/TSPD}) - P\left(\frac{(Tclear/TSPD) \cdot E_* \cdot I_t \cdot Pl}{Affinity + I_t + E_*}\right) \\
 V_{t+1} &= V_t + P(I_t \cdot 100/TSPD) - B(T_t, 1.0 - e^{-2.3/TSPD})
 \end{aligned}$$

$$\begin{aligned}
 N_{T_{t+1}} &= N_{T_t} - B\left(N_{T_t}, 1.0 - e^{-\frac{\frac{I_t \cdot Pl}{Affinity}}{1 + \frac{I_t \cdot Pl}{Affinity}} \cdot NtE/TSPD}\right) \\
 A_{T_{t+1}}^0 &= B\left(N_{T_t}, 1.0 - e^{-\frac{\frac{I_t \cdot Pl}{Affinity}}{1 + \frac{I_t \cdot Pl}{Affinity}} \cdot NtE/TSPD}\right) - B(A_{T_t}^0, 1.0 - e^{-0.6/TSPD}) \\
 A_{T_{t+1}}^j &= A_{T_t}^{j-1} - B(A_{T_t}^{j-1}, 1.0 - e^{-0.6/TSPD}) \\
 A_{T_{t+1}}^E &= A_{T_t}^{E-1} - B(A_{T_t}^{E-1}, 1.0 - e^{-0.6/TSPD})
 \end{aligned}$$

$$\begin{aligned}
E_{T_{A_{t+1}}}^0 &= A_{T_t}^E + E_{T_{A_t}}^0 - B(E_{T_{A_t}}^0, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^0, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - \\
&B(E_{T_{A_t}}^0, 0.02) \\
E_{T_{A_{t+1}}}^i &= E_{T_{A_t}}^i + 2 \cdot E_{T_{B_t}}^{i-1E} - B(E_{T_{A_t}}^i, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - \\
&B(E_{T_{A_t}}^i, 0.02) \\
E_{T_{A_{t+1}}}^E &= E_{T_{A_t}}^E + 2 \cdot E_{T_{B_t}}^{E-1E} - B(E_{T_{B_t}}^{E-1E}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^E, 0.02) \\
E_{T_{B_{t+1}}}^0 &= B(E_{T_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) \\
E_{T_{B_{t+1}}}^{il} &= E_{T_{B_t}}^{il-1} - B(E_{T_{B_t}}^{il-1}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{B_t}}^{il}, 0.02) \\
E_{T_{B_{t+1}}}^{iE} &= E_{T_{B_t}}^{iE-1} - B(E_{T_{B_t}}^{iE-1}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{B_t}}^{iE}, 0.02) \\
\\
W_{MT_{t+1}}^0 &= B(E_*, 0.02) \\
W_{MT_{t+1}}^m &= W_{MT_t}^{m-1} \\
W_{MT_{t+1}}^E &= W_{MT_t}^{E-1} \\
\\
M_{T_{t+1}} &= W_{MT_t}^E + M_{T_t} - B(M_{T_t}, 1.0 - e^{\frac{-\frac{I_t \cdot Pl}{Affinity}}{1 + \frac{I_t \cdot Pl}{Affinity}} \cdot NtE/TSPD}) \\
A_{MT_{t+1}}^0 &= B(M_{T_t}, 1.0 - e^{\frac{-\frac{I_t \cdot Pl}{Affinity}}{1 + \frac{I_t \cdot Pl}{Affinity}} \cdot NtE/TSPD}) - B(A_{MT_t}^0, 1.0 - e^{-0.4/TSPD}) \\
A_{MT_{t+1}}^k &= A_{MT_t}^{k-1} - B(A_{MT_t}^{k-1}, 1.0 - e^{-0.4/TSPD}) \\
A_{MT_{t+1}}^E &= A_{MT_t}^{E-1} - B(A_{MT_t}^{E-1}, 1.0 - e^{-0.4/TSPD}) \\
E_{MT_{A_{t+1}}}^0 &= A_{MT_t}^E + E_{MT_{A_t}}^0 - B(E_{MT_{A_t}}^0, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{MT_{A_t}}^0, 1.0 - \\
&e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^0, 0.02) \\
E_{MT_{A_{t+1}}}^i &= E_{MT_{A_t}}^i + 2 \cdot E_{MT_{B_t}}^{i-1E} - B(E_{MT_{A_t}}^i, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^i, 1.0 - \\
&e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{MT_{A_t}}^i, 0.02) \\
E_{MT_{A_{t+1}}}^E &= E_{MT_{A_t}}^E + 2 \cdot E_{MT_{B_t}}^{E-1E} - B(E_{MT_{A_t}}^E, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^E, 0.02) \\
E_{MT_{B_{t+1}}}^0 &= B(E_{MT_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) \\
E_{MT_{B_{t+1}}}^{il} &= E_{MT_{B_t}}^{il-1} - B(E_{MT_{B_t}}^{il-1}, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{B_t}}^{il}, 0.02) \\
E_{MT_{B_{t+1}}}^{iE} &= E_{MT_{B_t}}^{iE-1} - B(E_{MT_{B_t}}^{iE-1}, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{B_t}}^{iE}, 0.02)
\end{aligned}$$

A.3 With Exhaustion

Exhaustion happens when T-cells are over stimulated and die. When we add exhaustion to the Markov model of the system, a new value must be calculated which is the over stimulation of the T-cells. This over stimulation causes premature death of cells in Effector populations (5 times the over stimulation for still dividing cells and stimulation for end-cycle cells) and also death in cells converting to memory (stimulation).

$$\text{Stimulation (St)} \frac{\frac{I_t \cdot Pl}{Affinity}}{1 + \frac{I_t \cdot Pl}{Affinity}}$$

$$\text{Over-stimulation (OSt)} \frac{\frac{I_t \cdot Pl}{25 \cdot Affinity}}{1 + \frac{I_t \cdot Pl}{25 \cdot Affinity}}$$

$$E_{T_{A_t+1}}^0 = A_{T_t}^E + E_{T_{A_t}}^0 - B(E_{T_{A_t}}^0, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^0, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{T_{A_t}}^0, 0.02) - B(E_{T_{A_t}}^0, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{T_{A_t+1}}^i = E_{T_{A_t}}^i + 2 \cdot E_{T_{B_t}}^{i-1E} - B(E_{T_{A_t}}^i, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{T_{A_t}}^i, 0.02) - B(E_{T_{A_t}}^i, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{T_{A_t+1}}^E = E_{T_{A_t}}^E + 2 \cdot E_{T_{B_t}}^{E-1E} - B(E_{T_{B_t}}^{E-1E}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{A_t}}^E, 0.02) - B(E_{T_{A_t}}^E, 1.0 - e^{-1.0 \cdot St/TSPD})$$

$$E_{T_{B_t+1}}^{i0} = B(E_{T_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}})$$

$$E_{T_{B_t+1}}^{il} = E_{T_{B_t}}^{il-1} - B(E_{T_{B_t}}^{il-1}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{B_t}}^{il}, 0.02) - B(E_{T_{B_t}}^{il-1}, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{T_{B_t+1}}^{iE} = E_{T_{B_t}}^{iE-1} - B(E_{T_{B_t}}^{iE-1}, 1.0 - e^{-0.6/TSPD}) - B(E_{T_{B_t}}^{iE}, 0.02) - B(E_{T_{B_t}}^{il-1}, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$W_{MT_{t+1}}^0 = B(E_*, 0.02)$$

$$W_{MT_{t+1}}^m = W_{MT_t}^{m-1} - B(W_{MT_t}^{m-1}, 1.0 - e^{-1.0 \cdot St/TSPD})$$

$$W_{MT_{t+1}}^E = W_{MT_t}^{E-1} - B(W_{MT_t}^{E-1}, 1.0 - e^{-1.0 \cdot St/TSPD})$$

$$E_{MT_{A_t+1}}^0 = A_{MT_t}^E + E_{MT_{A_t}}^0 - B(E_{MT_{A_t}}^0, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{MT_{A_t}}^0, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^0, 0.02) - B(E_{MT_{A_t}}^0, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{MT_{A_t+1}}^i = E_{MT_{A_t}}^i + 2 \cdot E_{MT_{B_t}}^{i-1E} - B(E_{MT_{A_t}}^i, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}}) - B(E_{MT_{A_t}}^i, 0.02) - B(E_{MT_{A_t}}^i, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{MT_{A_t+1}}^E = E_{MT_{A_t}}^E + 2 \cdot E_{MT_{B_t}}^{E-1E} - B(E_{MT_{A_t}}^E, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{A_t}}^E, 0.02) - B(E_{MT_{A_t}}^E, 1.0 - e^{-1.0 \cdot St/TSPD})$$

$$E_{MT_{B_t+1}}^{i0} = B(E_{MT_{A_t}}^i, 1.0 - e^{\frac{-1.0}{((Ct-Bd) \cdot TSPH)}})$$

$$E_{MT_{B_t+1}}^{il} = E_{MT_{B_t}}^{il-1} - B(E_{MT_{B_t}}^{il-1}, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{B_t}}^{il}, 0.02) - B(E_{MT_{B_t}}^{il-1}, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

$$E_{MT_{B_t+1}}^{iE} = E_{MT_{B_t}}^{iE-1} - B(E_{MT_{B_t}}^{iE-1}, 1.0 - e^{-0.4/TSPD}) - B(E_{MT_{B_t}}^{iE}, 0.02) - B(E_{MT_{B_t}}^{iE}, 1.0 - e^{-5.0 \cdot OSt/TSPD})$$

B Markov Process

B.1 Introduction

We are representing the underlying Markov process from Chao's IS simulator. To do so, we are using the same populations as the Markov chain representation (App. A). The major modification comes in the time delay states which are now represented not by a succession on states, but by a exact time delay enforced by the scheduler. The affected states and their delays are: A_T Activated T-cell with a 19 hour delay (d_A), A_{MT} Activated Memory T-cell with a 1 hour delay (d_M), E_{TB}^i Effector T-cell generation i with a 5 hour delay (d_E), E_{MTB}^i Effector Memory T-cell generation i with a 5 hour delay (d_E) and W_{MT} Converting to Memory with a 14 day delay (d_W). When a cell enters one of these states, the entry time (t) is memorized and the exit time calculated from the entry time and the delay (d) is added to the scheduler. When the time equals an exit time on the scheduler, the cell is released and sent to the next state. The state S is thus defined as follows:

$$S = (T, I, V, N_T, M_T, A_T, A_{MT}, E_{TA}^i, E_{MTA}^i, E_{TB}^i, E_{MTB}^i, W_{MT})$$

$$E_* = \sum E_{TA}^i + \sum E_{MTA}^i + \sum E_{TB}^i + \sum E_{MTB}^i \forall i \in [0..18]$$

with $i = [0 \dots 18]$ number of times a T-Cell can divide

The transition rates enable us to calculate the probability of a rule being the next transition and its time. The time is an exponentially randomly distributed variable of the sum of all rates, and the probability of a rule being the next transition is its specific rate divided by the sum of all rates.

B.2 Parameters

Aff	= Match rule dependant	c	= 2.3 day^{-1}
λ	= $5 \cdot 10^4 \text{ day}^{-1}$	ρ	= $\frac{I}{Aff+I} \text{ day}^{-1}$
δ_T	= 0.1 day^{-1}	μ	= 0.02 day^{-1}
δ_I	= 0.8 day^{-1}	ϵ	= 24 day^{-1}
δ_E	= 0.6 day^{-1}	d_A	= 19 hours
δ_M	= 0.4 day^{-1}	d_M	= 1 hour
β	= $2 \cdot 10^{-7} \text{ day}^{-1}$	d_E	= 5 hours
π	= 100 day^{-1}	d_W	= 14 days
α	= $\frac{12}{Aff+I+E_*} \text{ day}^{-1}$		

B.3 Parallel equations

Origin	Parameter	Effect	Transition Rate
	$\xrightarrow{\lambda}$	$T + 1$	λ
T	$\xrightarrow{\delta_T}$	$T - 1$	$\delta_T \cdot T$
T	$\xrightarrow{\beta}$	$T - 1, I + 1$	$\beta \cdot V \cdot T$
I	$\xrightarrow{\delta_I}$	$I - 1$	$\delta_I \cdot I$
I	$\xrightarrow{\pi}$	$I, V + 1$	$\pi \cdot I$
I	$\xrightarrow{\alpha}$	$I - 1, E_*$	$\alpha \cdot I \cdot E_*$
V	\xrightarrow{c}	$V - 1$	$c \cdot V$
N	$\xrightarrow{\rho}$	$N - 1, A_T(t) + 1$	$\rho \cdot N$
A_T	$\xrightarrow{\delta_E}$	$A_T - 1$	$\delta_E \cdot A_T$
$A_T(t + d_A)$	\rightarrow	$A_T(t + d_A) - 1, E_{T_A}^0 + 1$	
$E_{T_A}^0$	$\xrightarrow{\delta_E}$	$E_{T_A}^0 - 1$	$\delta_E \cdot E_{T_A}^0$
$E_{T_A}^0$	$\xrightarrow{\mu}$	$E_{T_A}^0 - 1, W_{MT}(t) + 1$	$\mu \cdot E_{T_A}^0$
$E_{T_A}^0$	$\xrightarrow{\epsilon}$	$E_{T_A}^0 - 1, E_{T_B}^0(t) + 1$	$\epsilon \cdot E_{T_A}^0$
$E_{T_A}^i$	$\xrightarrow{\delta_E}$	$E_{T_A}^i - 1$	$\delta_E \cdot E_{T_A}^i$
$E_{T_A}^i$	$\xrightarrow{\epsilon}$	$E_{T_A}^i - 1, E_{T_B}^i(t) + 1$	$\epsilon \cdot E_{T_A}^i$
$E_{T_A}^i$	$\xrightarrow{\mu}$	$E_{T_A}^i - 1, W_{MT}(t) + 1$	$\mu \cdot E_{T_A}^i$
$E_{T_A}^E$	$\xrightarrow{\delta_E}$	$E_{T_A}^E - 1$	$\delta_E \cdot E_{T_A}^E$
$E_{T_A}^E$	$\xrightarrow{\mu}$	$E_{T_A}^E - 1, W_{MT}(t) + 1$	$\mu \cdot E_{T_A}^E$
$E_{T_B}^i$	$\xrightarrow{\delta_E}$	$E_{T_B}^i - 1$	$\delta_E \cdot E_{T_B}^i$
$E_{T_B}^i$	$\xrightarrow{\mu}$	$E_{T_B}^i - 1, W_{MT}(t) + 1$	$\mu \cdot E_{T_B}^i$
$E_{T_B}^i(t + d_E)$	\rightarrow	$E_{T_B}^i(t + d_E) - 1, E_{T_A}^{i+1} + 1$	
$W_{MT}(t + d_W)$	\rightarrow	$W_{MT}(t + d_W) - 1, M_T + 1$	
M_T	$\xrightarrow{\rho}$	$M_T - 1, A_{MT}(t) + 1$	$\rho \cdot M_T$
A_{MT}	$\xrightarrow{\delta_M}$	$A_{MT} - 1$	$\delta_M \cdot A_{MT}$
$A_{MT}(t + d_M)$	\rightarrow	$A_{MT}(t + d_M) - 1, E_{MT_A}^0 + 1$	
$E_{MT_A}^0$	$\xrightarrow{\delta_M}$	$E_{MT_A}^0 - 1$	$\delta_M \cdot E_{MT_A}^0$
$E_{MT_A}^0$	$\xrightarrow{\mu}$	$E_{MT_A}^0 - 1, W_{MT}(t) + 1$	$\mu \cdot E_{MT_A}^0$
$E_{MT_A}^0$	$\xrightarrow{\epsilon}$	$E_{MT_A}^0 - 1, E_{MT_B}^0(t) + 1$	$\epsilon \cdot E_{MT_A}^0$
$E_{MT_A}^i$	$\xrightarrow{\delta_M}$	$E_{MT_A}^i - 1$	$\delta_M \cdot E_{MT_A}^i$
$E_{MT_A}^i$	$\xrightarrow{\epsilon}$	$E_{MT_A}^i - 1, E_{MT_B}^i(t) + 1$	$\epsilon \cdot E_{MT_A}^i$
$E_{MT_A}^i$	$\xrightarrow{\mu}$	$E_{MT_A}^i - 1, W_{MT}(t) + 1$	$\mu \cdot E_{MT_A}^i$
$E_{MT_A}^E$	$\xrightarrow{\delta_M}$	$E_{MT_A}^E - 1$	$\delta_M \cdot E_{MT_A}^E$
$E_{MT_A}^E$	$\xrightarrow{\mu}$	$E_{MT_A}^E - 1, W_{MT}(t) + 1$	$\mu \cdot E_{MT_A}^E$
$E_{MT_B}^i$	$\xrightarrow{\delta_E}$	$E_{MT_B}^i - 1$	$\delta_E \cdot E_{MT_B}^i$
$E_{MT_B}^i$	$\xrightarrow{\mu}$	$E_{MT_B}^i - 1, W_{MT}(t) + 1$	$\mu \cdot E_{MT_B}^i$
$E_{MT_B}^i(t + d_E)$	\rightarrow	$E_{MT_B}^i(t + d_E) - 1, E_{MT_A}^{i+1} + 1$	

C NI simulator without T-cell interaction

```
/*
 * SimpleModel.java
 *
 * Created on February 11, 2005, 6:57 PM
 */

import java.util.*;
import java.io.*;
import java.lang.Math;

import cern.jet.random.*;
import cern.jet.random.engine.*;
/**
 *
 * @author ewinning
 */
public class SimpleModel {

    public int virus;
    public int target;
    public int infected;

    public int virus_new;
    public int target_new;
    public int infected_new;

    public int tInfected;

    // time is in minutes => 20 days = 480 hours = 28800 minutes
    public int time = 0;
    public final int timestep = 1;
    public final int tsph = 60/timestep;
    public final int tspd = 24 * tsph;

    public final double lambda = 50000.0;
    public final double delta_t = 0.01;
    public final double beta = 2e-7;
    public final double delta_i = 0.7;
    public final double pi = 100.0;
    public final double c = 2.3;

    public final String symbol = "□";

    public MersenneTwister mt;
    public Binomial bin;
    public Poisson poi;

    /** Creates a new instance of SimpleModel */
    public SimpleModel() {
```

```

mt = new MersenneTwister(new Date());
bin = new Binomial(100, 0.1, mt);
poi = new Poisson(100, mt);

    virus = 50;
    infected = 0;
    target = (int)Math.pow(10,6);

    for(time = 0; time < (20 * tspd); time++){
        if (time % tsph == 0) System.out.println((double)time/tspd +
            symbol + target + symbol + infected + symbol + virus);

        tInfected = b(target, 1.0-Math.exp(-beta*virus/tspd));
        target_new = target + p(lambda/tspd) -
            b(target, 1.0-Math.exp(-delta_t/tspd)) - tInfected;
        infected_new = infected + tInfected -
            b(infected, 1.0-Math.exp(-delta_i/tspd));
        virus_new = virus + p((pi/tspd)*infected) -
            b(virus, 1.0-Math.exp(-c/tspd));

        target = target_new;
        virus = virus_new;
        infected = infected_new;
    }
    System.out.println((double)time/tspd + symbol + target + symbol +
        infected + symbol + virus);
}

private int p(double n){
    if (n != 0)
        return poi.nextInt(n);
    return 0;
}

private int b(int n, double p){
    if (n==0) return 0;
    if (p==0.0) return 0;

    return bin.nextInt(n, p);
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {

    SimpleModel sm = new SimpleModel();
}
}

```

D Simulator initialization sequence

The Initialization sequence from Driver.java is where the whole simulator is set up. The following sequence is observed:

1. define Match rule for the program (default is Hamming)
2. define and initialize the Random number generator
3. Selection of the Match rule
 - (a) Set Alphabet size and peptide length
 - (b) Set number of T-cell clones (i.e. number of different TCRs in the body)
 - (c) Set MHC length
4. Create Antigen, Self and T-cell array lists
5. Create Random MHC String (Depends on Constants.NUMMHC for number of MHC) (L143)
6. Create Self chains according to Constants.NUMSELF and assign them to MHCs
7. Create the virus
 - (a) Define and randomize antigen string
 - (b) Assign Antigen string to a MHC
 - (c) Create Antigen object with complex = antigen string + MHC
8. Calculate the distances (between antigen complex and self complex)
9. Create T-cells
 - (a) Create random strings at a certain distance using lazy generator. For each antigen, create a random number of clones at the cross reactivity distance (ie. from 0 to Negative Cutoff distance). (TCRGenerator L95)
 - (b) Make Thymus selection and reactivity selection. If it survives add it to the list of survivors (TCRGenerator L153)
 - (c) Use ths surviving chains to Create each T-cell object
10. Start Simulation